What Is Claimed Is:

1. A method to efficiently realize class initialization barriers in a multitasking virtual machine, wherein class loading always takes place before class initialization, and wherein a class initialization barrier guarantees that a class is initialized before the class is first used by a program, comprising:

associating a shared runtime representation of the class with a task class mirror table that comprises at least one entry per-task, including an initialized entry, for a plurality of tasks, wherein each entry holds either a null pointer value or a non-null pointer to a task class mirror object, wherein all entries of a task mirror table that hold a non-null pointer value and that are associated with a same task hold a pointer to a same task class mirror object, wherein the task class mirror object holds a task private representation of the class for that task;

using the initialized entry of a task in the task class mirror table to determine whether this task has initialized the class associated with the task class mirror table; and

accessing the task class mirror object associated to a particular task.

2. The method of claim 1,

wherein each task is associated with a unique integer value;

wherein the unique integer value is used to compute a byte-offset from a beginning of task class mirror tables that can be used to retrieve from the initialized entry of any task class mirror table the pointer to the task class mirror object; and

wherein a computed byte-offset to the initialized entry is stored in a descriptor of a plurality of threads executing on behalf of a corresponding task.

3. The method of claim 2, further comprising:

creating the task class mirror table and associating the task class mirror table with the shared runtime representation of the class upon creation of the shared runtime representation of the class; and

setting all entries of the task class mirror table to the null pointer value.

4. The method of claim 3, further comprising:

examining the initialized entry of the task in the task class mirror table associated with the class in order to determine if that task has initialized the class, wherein the byte-offset to the initialized entry from the beginning of the task class mirror table is obtained from the descriptor of a thread performing an examination on behalf of the task; and

initializing the class by the task if the class is not already initialized, wherein a null pointer stored at the initialized entry indicates that the class has not initialized the task.

5. The method of claim 4, further comprising:

upon completion of initialization of the class by the task, setting the initialized entry of the task class mirror table associated with the class to the task class mirror object that holds a representation of the class that is private to the task; and

setting this task class mirror object to a fully initialized state.

6. The method of claim 5, wherein task class mirror tables associated with classes that have a non-empty initialization function includes one resolved entry per-task in addition to one initialized entry per-task, for the plurality of tasks.

7. The method of claim 6, wherein task class mirror tables associated with classes that have an empty initialization function include one resolved entry per-task in addition to an initialized entry per-task, for the plurality of tasks.

8. The method of claim 7, further comprising:

upon loading any class by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to loaded; and

assigning the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

9. The method of claim 8,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are consecutive; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of the pointer to the task class mirror object.

10. The method of claim 8,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.